Experiment No 09 Image Enhancement -02

### **Name: Rebecca Dias Roll No.: 19 Class: BE CMPN A PID: 182027**

# Aim:

To perform the image enhancement techniques of Contrast Stretching, Bit Plane Slicing and Dynamic Range Compression of an image signal in Python

# Theory:

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. Image Enhancement can be done in two domains: Spatial Domain and Frequency Domain. Spatial domain enhancement involves direct manipulation of pixels of the image. Spatial domain enhancement can be carried out in two different ways: point processing and neighborhood processing.

# Point Processing:

Point processing uses only the information in individual pixels to produce new images. A transform may be computed on the basis of regional or global information and then applied to the individual points.

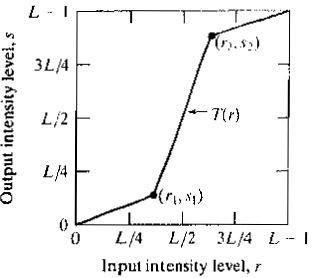
Some of the point processing techniques are Digital

* Negative
* Thresholding
* Gray Level Slicing
* Contrast Stretching Bit plane Slicing
* Dynamic Range Compression
* Power Law Transformation

# Contrast Stretching

Low contrast images can result from poor illumination, lack of dynamic range in the sensor or wrong setting of lens aperture during image acquisition. The idea behind contrast stretching is to increase the contrast of the image by making the dark portion darker and bright portion brighter. It can only apply a linear scaling function to the image pixel values. Thus contrast stretching is a process that expands the range of intensity levels in an image so that it spans the full intensity range of the recording medium or display device.

A typical transformation used for contrast stretching is given below:



The location of the points (r1, s1) and (r2, s2) control the shape of the transformation function. If r1=s1 and r2=s2, the transformation is identity transformation. If r1=r2, s1=0and s2=L-1, the transformation is a thresholding function that creates a binary image. The intermediate values of (r1, s1) and (r2, s2) produce various degrees of spread in the intensity levels of the output image, thus altering its contrast. In general, r1≤ r2 and s1≤ s2 is assumed so that the function is single valued and monotonically increasing. This condition preserves the order of intensity levels.

Example:

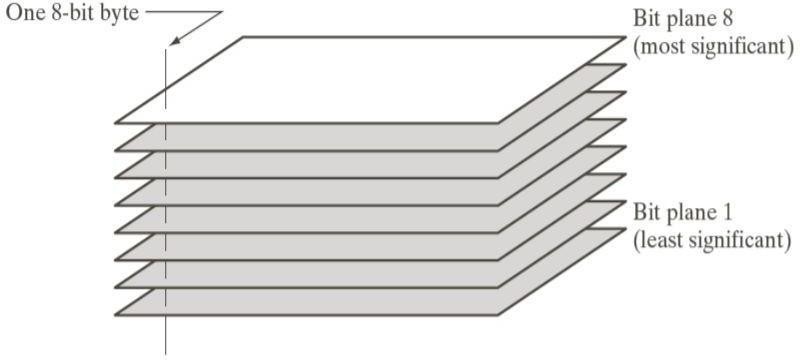


Low Contrast Image Contrast Stretched Image

# Bit plane Slicing

Pixels are digital numbers, each one composed of bits. Instead of highlighting gray-level range, we could highlight the contribution made by each bit. This method is useful in image compression. Most significant bits contain the majority of visually significant data.

For an 8-bit gray level image, the image is composed of 8 1-bit planes as shown below:



Plane 0 contains the least significant bit and plane 7 contains the most significant bit. In terms of 8-bits bytes, plane 0 contains all lowest order bits in the bytes comprising the pixels in the image and plane 7 contains all high order bits.



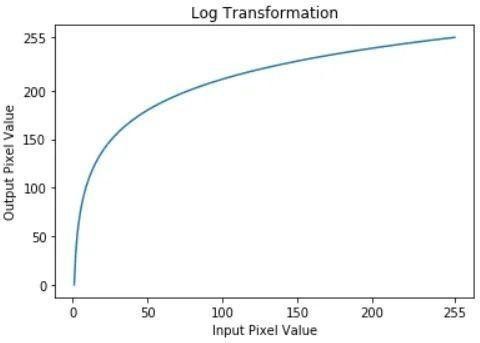
Example: Original and Bit Plane 0-7

# Dynamic Range Compression (Log Transformation)

At times, the dynamic range of image exceeds the capability of the display device. In image some pixel values are so large that the other low value pixels get obscured. We need a technique to see even low value pixels. The technique to compress the dynamic range is known as dynamic range compression. The function used is log transformation function.

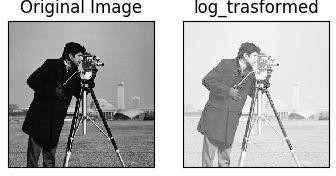
The log transformations can be defined by this formula: **s = c log(r + 1).**

Where s and r are the pixel values of the output and the input image and c is a constant.



The transformation function is given by

During log transformation, the dark pixels in an image are expanded as compared to the higher pixel values. The higher pixel values are kind of compressed in log transformation. This results in the following image enhancement.



The value of c in the log transform adjust the kind of enhancement you are looking for

# Programming Exercise

1. Do thresholding of a random image at T=170 using contrast stretching program.
2. Do contrast stretching of a random image with (r1,s1)=(65,90) and (r2,s2)=(165,190)
3. Do bit plane slicing of a random image and combine bitplane 8,7 and 6
4. Do log transformation of a random image with c as your roll number.

## **1.Do thresholding of a random image at T=170 using contrast stretching program.**

|  |
| --- |
| import numpy as np  import pandas as pd  import cv2  from google.colab.patches import cv2\_imshow  from skimage import io  from PIL import Image  import matplotlib.pyplot as plt  img = cv2.imread('/content/image.jpg')  original = img.copy()  xp = [0, 170]  fp = [0, 255]  x = np.arange(256)  table = np.interp(x, xp, fp).astype('uint8')  img = cv2.LUT(img, table)  cv2\_imshow(original) |



# 2.Do contrast stretching of a random image with (r1,s1)=(65,90) and (r2,s2)=(165,190)

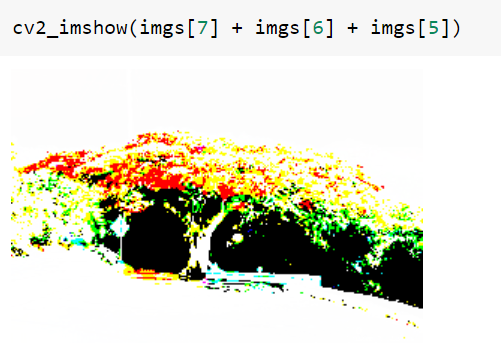
|  |
| --- |
| # Function to map each intensity level to output intensity level.  def pixelVal(pix, r1, s1, r2, s2):      if (0 <= pix and pix <= r1):          return (s1 / r1)\*pix      elif (r1 < pix and pix <= r2):          return ((s2 - s1)/(r2 - r1)) \* (pix - r1) + s1      else:          return ((255 - s2)/(255 - r2)) \* (pix - r2) + s2    # Open the image.  img = cv2.imread('/content/image.jpg')    # Define parameters.  r1,r2,s1,s2 = 65,165,90,190  # Vectorize the function to apply it to each value in the Numpy array.  pixelVal\_vec = np.vectorize(pixelVal)    # Apply contrast stretching.  contrast\_stretched = pixelVal\_vec(img, r1, s1, r2, s2)    # Save edited image.  cv2.imwrite('contrast\_stretch.jpg', contrast\_stretched)  cv2\_imshow(img)  cv2\_imshow(contrast\_stretched) |

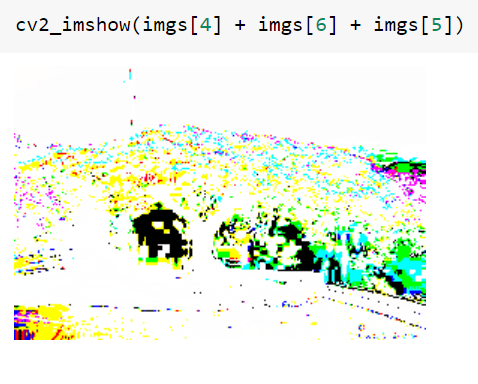


## **3.Do bit plane slicing of a random image and combine bitplane 8,7 and 6**

|  |
| --- |
| imgs = [255 \* ((img & (1 << i)) >> i) for i in range(8)]  print("Bit Planes 1 to 8:")  cv2\_imshow(cv2.hconcat(imgs)) |







## **4.Do log transformation of a random image with c as your roll number.**

|  |
| --- |
| # Open the image.  img = cv2.imread('/content/image.jpg')  # Apply log transform.  # c = 255/(np.log(1 + 19))  c=19  log\_transformed = c \* np.log(1 + img)  # Specify the data type.  log\_transformed = np.array(log\_transformed, dtype = np.uint8)    # Save the output.  cv2\_imshow(img)  cv2\_imshow(log\_transformed) |



# Conclusion:

Studied image enhancement techniques of contrast stretching, bit plane slicing and dynamic range compression of an image signal. Implemented it using the python language and the output was accurately obtained as well.